# NAVAL POSTGRADUATE SCHOOL
# MONTEREY, CALIFORNIA


# THESIS

**ROUTE OPTIMIZATION MODEL FOR STRIKE AIRCRAFT**

by

Steve H. K. Lee

September, 1995

Thesis Advisor: Kevin Wood

**Approved for public release; distribution is unlimited.**

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE<br>September 1995. | 3. REPORT TYPE AND DATES COVERED<br>Master's Thesis |
|---|---|---|

| 4. TITLE AND SUBTITLE ROUTE OPTIMIZATION MODEL FOR STRIKE AIRCRAFT | 5. FUNDING NUMBERS |
|---|---|
| 6. AUTHOR(S) Steve H. K. Lee | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Postgraduate School<br>Monterey CA 93943-5000 | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

| 12a. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution is unlimited. | 12b. DISTRIBUTION CODE |
|---|---|

13. ABSTRACT *(maximum 200 words)*

A model is designed and implemented to construct a "flyable," least-risk route for strike aircraft from takeoff to target, through enemy radars, in a defined area of operations. A network is first constructed by discretizing the airspace into a three-dimensional grid of nodes and then connecting adjacent nodes with arcs. A shortest-path model in this network is then constructed with arc lengths that are a function of the probability of detection by radars monitoring the area of operations. A side constraint on fuel consumption ensures that routes are feasible. Lagrangian relaxation is used to incorporate this constraint into the problem and a shortest-path algorithm solves a sequence of shortest-path sub-problems to obtain a near-optimal route.

AROMA (Automatic Route Optimization Model for Aircraft) is implemented in C++ on a Silicon Graphics Onyx computer with 192 megabytes of memory. Test problems comprising 240,000 nodes and more than 2 million arcs are used to evaluate the model. Realistic routes are generated in approximately 2 to 3 minutes. A graphical interface displays the routes and facilitates interactive analysis and model evaluation.

| 14. SUBJECT TERMS Optimization, Langragian relaxation, Route planning, Flight planning, Routing, Shortest-path. | | | 15. NUMBER OF PAGES 74<br>16. PRICE CODE |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT<br>Unclassified | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>Unclassified | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>Unclassified | 20. LIMITATION OF ABSTRACT<br>UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

**Approved for public release; distribution is unlimited.**

# ROUTE OPTIMIZATION MODEL FOR STRIKE AIRCRAFT

Steve H. K. Lee
Captain, Republic of Singapore Airforce
B.Eng., University of Singapore, 1988

Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 1995**

Author: _____
Steve H. K. Lee

Approved by: _____
R. Kevin Wood, Thesis Advisor

_____
Gordon H. Bradley, Second Reader

_____
Frank C. Petho, Acting Chairman
Department of Operations Research

**ABSTRACT**

A model is designed and implemented to construct a "flyable," least-risk route for strike aircraft from takeoff to target, through enemy radars, in a defined area of operations. A network is first constructed by discretizing the airspace into a three-dimensional grid of nodes and then connecting adjacent nodes with arcs. A shortest-path model in this network is then constructed with arc lengths that are a function of the probability of detection by radars monitoring the area of operations. A side constraint on fuel consumption ensures that routes are feasible. Lagrangian relaxation is used to incorporate this constraint into the problem and a shortest-path algorithm solves a sequence of shortest-path sub-problems to obtain a near-optimal route.

AROMA (Automatic Route Optimization Model for Aircraft) is implemented in C++ on a Silicon Graphics Onyx computer with 192 megabytes of memory. Test problems comprising 240,000 nodes and more than 2 million arcs are used to evaluate the model. Realistic routes are generated in approximately 2 to 3 minutes. A graphical interface displays the routes and facilitates interactive analysis and model evaluation.

**TABLE OF CONTENTS**

## LIST OF FIGURES

## LIST OF TABLES

## EXECUTIVE SUMMARY

For an air force, one of the main problems of mission planning is the selection of a route that a strike aircraft can use to reach a target from a base while flying through a group of enemy radars. At the tactical level, a solution to this problem would help pilots select "safe" potential routes and identify areas of enemy weakness prior to a mission. At the operational level, a solution would help planners derive "realistic" routes for decision-making purposes such as force-structuring and campaign analysis.

This thesis designs and implements a prototypic Automatic Route Optimization Model for Aircraft (AROMA). The model computes, approximately, a least risk route that a strike aircraft could use to reach a target from a base while flying through enemy radar coverage. The main considerations in this model are detection by the enemy radars, fuel consumption characteristics of the aircraft and the terrain in the area of operations.

AROMA is formulated as a constrained shortest-path model. The model discretizes the airspace into a three-dimensional grid of nodes and then connects adjacent nodes with arcs that represent potential flight segments. A shortest-path model in this network is then constructed with arc lengths that are a function of the probability of detection by radars monitoring the area of operations.

Radar detection, for any individual radar, is modelled as a region of constant detection probability. That is, the probablity of detection attached to a unit flight segment in the region is a given constant. Probablistic assumptions about the independence of detection events between flight segments and between different radars are also made. A fuel consumption constraint is introduced to ensure that routes are feasible. This is translated into a simplified fuel consumption model that accounts for different altitudes and profiles that an aircraft can fly. Terrain is also factored into the model so that the aircraft avoids colliding into obstructions.

Because AROMA models a computationally difficult (NP-complete) problem, Lagrangian relaxation is used to incorporate the fuel constraint into the problem and a label-correcting shortest-path algorithm solves a sequence of relaxed shortest-path sub-problems to obtain a near-optimal route. The solution algorithm is coded in C++ and runs on a Silicon Graphics (SGI) computer.

Test cases for areas of up to 200 square nautical miles (at intervals of 1 nautical mile) and six allowable height levels were evaluated. These problems comprise 240,000 nodes and more than 2 million arcs. Realistic routes are generated in approximately 2 to 3 minutes.

A graphical evaluation suite (GES) was also developed for the modeller to visually evaluate the solution routes produced

AROMA. This suite allows routes generated to be displayed, and has features that permit interactive analysis of the algorithm, for instance, by allowing the algorithm to be run in a step-by-step mode. GES proved to be useful in analyzing how the algorithm converges to a solution, and in evaluating the "goodness" of solution routes.

Four conclusions can be made after the evaluation of the model. First, there can exist a gap between a "good" solution and its theoretical lower bound. Second, solution quality generally does not improve after 15 to 20 iterations for the test problems, and the time taken for each successive iteration tends to increase. Therefore, some fixed limit for the number of iterations, say 20, is probably sufficient for most similar problems. Detection "costs" were computed using negative logarithms of non-detection probablities. However, the third conclusion is that simply using detection probablities as arc costs yields very similar solutions. Finally, due to discretization of the airspace, errors in the computation of fuel consumption are introduced into the problem, which directly affect the feasibility of solutions (giving fewer possible routes). Unlike other approximations, discretization is highly visible since it results in a jagged route with unnecessary turns. Post-processing smoothing is suggested to alleviate this effect.

## THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

# I. INTRODUCTION

This thesis describes the design and implementation of a prototypic automatic route optimization model for aircraft mission planners. This model is called AROMA (Automatic Route Optimization Model for Aircraft).

## A. BACKGROUND

Military mission planning at the tactical and operational level is becoming increasingly complex. To help planners and decision-makers overcome some of the problems of planning, automated (computerized) systems have become popular.

For an air force, one of the main problems of mission planning is the selection of a route that a strike aircraft can use to reach a target from a base while flying through enemy radar coverage. At the tactical level, a solution to this problem would help pilots select "safe" potential routes and identify areas of weak enemy radar coverage prior to a mission. At the operational level, a solution would help planners derive "realistic" routes for decision-making purposes such as force-structuring and campaign analysis. The incorporation of an automatic routing model into air mission planning systems would therefore be highly beneficial.

## B. PROBLEM DESCRIPTION

### 1. Overview

Route optimization provides an air planner the "best" route to take to attack a target and to return to base. For simplicity, only the attack leg of the route, from takeoff to target, is considered in this thesis. An optimal solution to this route planning problem will take into consideration the terrain, aircraft fuel consumption characteristics, enemy radar capabilities and general air planning doctrine. The generated output is a series of waypoints (cartesian coordinates) describing the profile of a route that approximately minimizes the risk of detection by enemy radars subject to fuel consumption constraints on the route.

Three principal areas of development are identified as requirements for AROMA. First, the optimization algorithm must be able to compute near-optimal routes for realistically sized problems in a few minutes. Second, a decomposition technique (Lagrangian relaxation) must be employed to achieve quick solutions. Third, a graphical interface is needed to display the routes and facilitate interactive analysis and model evaluation. These three requirements were fulfilled in the development of AROMA.

### 2. Data Requirements

AROMA, like most route planning systems use some form of digitally encoded terrain information data (Leary

1994). The format to be used for AROMA is DTED (Digital Terrain Elevation Data) which is the current standard format issued by DMA (Defense Mapping Agency). The basic information encoded in the data is the terrain height of equally spaced grid points for an area of operation. This information can be used to help determine enemy radar detection probabilities, line-of-sight for radars, and terrain avoidance by the aircraft. Only terrain avoidance is currently implemented in AROMA.

General fuel consumption characteristics and aircraft maneuver capabilities are needed to constrain the routes generated by any route planning model. A realistic but simplified model is used in AROMA. Fuel consumption rates along flight segments are assigned linear costs depending on the aircraft's height, profile (climbing, diving, level flight) and the length of that segment.

In the design of AROMA, it is assumed that intelligence on the area of operations includes terrain maps and the location and performance characteristics of all enemy radars. Enemy radar information is required to compute probability of detection based on geometry, terrain and aircraft characteristics.

## 3. Optimization Model

AROMA computes a flyable, approximately least-risk route for strike aircraft from takeoff to target in a defined

area of operations. The return route from the target to landing at the home base is not considered in this model. The MOE (measure of effectiveness) to be approximately minimized is the aggregate probability of detection by enemy radars. The main constraints of the model pertain to maintaining feasible routes both in terms of fuel and aircraft operating characteristics. Terrain avoidance is also considered, to avoid solutions that would cause the aircraft to crash into terrain.

#### 4. Implementation

The performance requirements for the model are that it must generate solutions for typical problems of 200 square nautical miles, in a reasonable amount of time, i.e., in a few minutes. An area of 200 square nautical miles covers the range of several typical strike aircraft.

In order to meet performance requirements, the model is implemented on a Silicon Graphics Onyx computer with 192 megabytes of memory. In addition, a Graphical Evaluation Suite (GES) is constructed to display the routes and facilitate interactive analysis and model evaluation.

## C. LITERATURE SURVEY

The general methodology of this thesis is to discretize the search space into a three-dimensional grid, connecting adjacent nodes with arcs, assign some form of arc costs and

determine the least cost route from the start point to the designated goal. This methodology has been used elsewhere.

Leary (1994) models a helicopter route optimization problem that considers the minimization of detection by radars, without considering aircraft fuel constraints. However, a large part of Leary's work pertaining to the radar detection model is useful for this thesis. One of Leary's recommendations is that a side constraint on fuel consumption be incorporated into the problem using the Lagrangian relaxation.

Boerman (1994) models a shortest-path AUV (Autonomous Underwater Vehicle) route through a mapped minefield. A side constraint on shock acceptable to the AUV is considered. A Lagrange multiplier $\mu$ is used to incorporate the constraint into the objective function. However, determination of an optimal value of $\mu$ is done experimentally. Boerman concludes that an automated visual means of analyzing routes would be useful for analysis of solutions.

Ong (1990) models a AUV obstacle avoidance problem. Conceptually, this is equivalent to terrain avoidance by the aircraft modelled in this thesis. Wrenn's (1989) work on fuel consumption modelling for cruise-missile route planning provides some basic foundation for developing our fuel consumption model.

## D. ORGANIZATION

The remainder of this thesis explains how AROMA is developed, implemented and evaluated. Chapter II describes the general solution methodology, the mathematical formulation of AROMA, data requirements and construction of the network model. Chapter III explains the implementation of the model. This explanation covers the programming environment, data structures and algorithms used. Chapter IV evaluates the results and performance of the model. Finally, Chapter V states some important conclusions and observations, and recommends areas for future research.

# II. DEVELOPMENT OF THE MODEL

The key stages in developing AROMA are described in this chapter. First, the route optimization problem is defined and a mathematical representation of this problem is formulated. Next, an appropriate solution methodology is outlined. Then, data requirements are identified and requisite sub-models are developed. Finally, the network structure required for AROMA is developed in detail.

## A. GENERAL PROBLEM STATEMENT

The aircraft route optimization model to be developed in this thesis computes, approximately, a least risk route that a strike aircraft could use to reach a target from a base while flying through enemy radars, subject to a fuel consumption constraint. This can be formulated as a *constrained shortest-path model*.

The constrained shortest-path model is an integer programming problem which is NP complete (Ahuja, et al., 1993, pp. 600-601). This thesis will not attempt to solve the constrained shortest-path problem directly, but will use a decomposition strategy (see Section IIC) to compute a near-optimal solution.

## B. MATHEMATICAL FORMULATION

The key purpose of this formulation is to elucidate the route optimization problem and its associated data concisely. Section IID will elaborate on the data requirements.

**Model name:** The constrained shortest-path problem

**Indices:**

$i,j \in N$ : Network nodes
$(i,j) \in A$ : Network arcs

Note: In this formulation, it is assumed that the aircraft starts at node 1 and the target is at node n.

**Data:**

$c_{ij}$ : Detection "cost" of traversing arc (i,j).

$t_{ij}$ : Fuel "cost" of traversing arc (i,j).

T : Maximum fuel available for a mission.

**Variables:**

$x_{ij}$ : binary variable that is 1 if arc (i,j) is in the optimal path, and 0 otherwise.

**Formulation:**

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij}$$

subject to

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 \text{ for } i = 1 \\ 0 \text{ for } i \in N - \{1,n\} \\ -1 \text{ for } i = n \end{cases}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \le T$$

$$x_{ij} \in \{0,1\} \text{ for all } (i,j) \in A$$

## C. GENERAL SOLUTION METHODOLOGY

An overview of the solution methodology is given below, and is elaborated in Chapter III.

A network is first constructed by discretizing the airspace into a three-dimensional grid of nodes and then connecting adjacent nodes with arcs. A shortest-path model in this network is then constructed with arc lengths that are a function of the probability of detection by radars monitoring the area of operations. A side constraint on fuel consumption is introduced to ensure that routes are feasible.

At this point, the model is a constrained shortest path problem as described in the previous section. However, because direct solution of the model would be difficult, Lagrangian relaxation is used to incorporate the fuel constraint into the objective function. Then, a fast label-correcting shortest-path algorithm solves a sequence of relaxed shortest-path sub-problems to obtain a near-optimal route.

## D. COMPONENT MODELS AND DATA REQUIREMENTS

### 1. Terrain Model

#### a. Purpose

A terrain model for AROMA is required for two main purposes. First, aircraft should avoid terrain. This implies that feasible routes must not pass through terrain obstructions. The second purpose, which has not been modelled in this thesis, is to allow for the incorporation of radar line-of-sight calculation into the model. Radar line-of-sight calculations should be added to account for the inability of radar to detect a target masked by terrain.

#### b. DTED Format

DTED is used to represent terrain data mainly because it is commonly used in most mission-planning models, and also because it is easy to manipulate. A single DTED file is a digitized representation of ground elevations of a region with dimensions of one degree longitude by one degree latitude (about 60 nautical miles). Each DTED file has about 16 megabytes of terrain elevation data giving elevations at intervals of approximately 100 meters, as well as other "housekeeping" information such as the location of the grid square.

### c. Data Requirements

Terrain data suitable for AROMA must be generated. Since the area of operations for the model developed here is defined as 200 square nautical miles, more than one DTED file is required to specify the terrain for the model. It was decided that each grid square in the terrain model should be one nautical mile because it can be reasonably assumed that aircraft would not change headings or levels within one nautical mile (at typical speeds of 400 nautical miles per hour, one nautical mile equals about nine seconds). Also, one nautical mile provides reasonable resolution for the radar detection model (to be described later).

To generate a 200 square nautical mile terrain file with a grid interval of one nautical mile, several DTED files must be processed together. This pre-processing is done only when areas of operation change. Otherwise, the same terrain file can be used repeatedly. Each terrain file used by the model contains about five megabytes of data.

## 2. Fuel Consumption Computations

### a. Purpose

Fuel consumption is modelled to ensure that routes generated are feasible in terms of the fuel capacity of the aircraft. In flight planning, fuel consumption rates are used to compute the amount of fuel required for different profiles such as climbing, high-level or low-level flight.

Each aircraft can carry different configurations of fuel and loads which will affect these rates, and thus the mission ranges.

### b. Data Requirements

For the purpose of this thesis, fuel consumption for a fictitious aircraft with a fixed load and fuel capacity (T) is used. These rates (in pounds of fuel per minute) pertain to different flight levels as well as climbing and diving profiles for a particular speed. Based on this speed, the fuel consumption rates are re-scaled from pounds per minute to pounds per nautical mile. Subsequently, fuel costs $t_{ij}$, for each arc in the network model, can be computed based on the product of these rates and the Euclidean length of that arc. In general, higher flight altitudes require less fuel than lower ones and a dive profile consumes less fuel than a climb.

It is recognized that an even more sophisticated modelling of fuel consumption rates is possible, but in consultation with mission planners, it was agreed that the current approach is a reasonable approximation.

## 3. Radar Detection Model

### a. Description

To compute the risk of a route to the aircraft and pilot, the detection probability of all radars monitoring

that route must be quantified. AROMA uses a radar model commonly used in manual mission planning that employs a hemispherical region of constant radar detection probability. This means that each flight segment that is within the detection region of a radar will be assigned a fixed detection probability.

### b. Arc Cost Assignment and Assumptions

In the assignment of arc costs, $c_{ij}$, two key assumptions have been made. First, each radar detection event (along each arc) and between different radars is assumed to be independent. This allows the model to properly aggregate the detection costs, according to the procedure specified by Leary (1994, pp. 23). This involves the use of logarithms to linearize the detection costs. The aggregated probablity of non-detection on an arc is

$$Q_{ij} = \prod_{r \epsilon R} (1 - P^{r}_{ij})$$

where R is the set of all enemy radars and $P^{r}_{ij}$ is the probablity a radar r will detect an aircraft traversing arc (i,j). Since we seek to maximize the probability of non-detection, our objective function could be:

$$\text{maximize} \prod_{(i,j) \epsilon A} Q_{ij}^{xij}$$

Leary shows this objective will yield the same solution as

$$\text{minimize} \sum_{(i,j) \epsilon A} -\log(Q_{ij}) x_{ij}$$

Therefore,

$$c_{ij} = -\log\left( \prod_{r \epsilon R} (1 - P^{r}_{ij}) \right)$$

may be used as the detection cost on arc(i,j).

The second assumption is that the detection cost for an arc is that of the node from which it originates. Little effort would be required to modify this to be the average of the detection probabilities of the two nodes at the head and tail of the arc, or some more accurate approximation of reality.

### c. Data requirements

Currently, the essential data for each radar is its location, maximum and minimum range and probability of detection ($P_d$). The maximum range is the radius of the region within the $P_d$ is constant. The minimum range allows for close-in "blind regions" for each radar. More sophisticated radar modelling like that described by Leary (1994) could be incorporated in AROMA without any major changes to the model. This is true because the assignment of "probability mass" to each arc of the network model is done as a pre-processing step, and the optimization model simply reads this mass and assigns the corresponding arc costs, $c_{ij}$, in the network. Therefore, even if a more sophisticated radar model is used, only minor changes have to be made to the pre-processing stage and no changes made to the optimization model itself.

## E. NETWORK

### 1. Construction

A network representation of the area of operation (200 square nautical miles) is constructed using a three dimensional lattice structure. Each co-planar grid point is one nautical mile apart, as explained earlier. The altitude separation used (up to 6 levels in our model) is in intervals of 300 meters starting at 100 meters. This covers typical attack ingress altitudes, which are normally low (less than 2000 meters) to avoid radar detection.

The network is constructed by joining each lattice node to its nearest neighbors. There are 24 nearest neighbors to each node except for those nodes on the perimeter of the network. The nearest neighbors are geometrically defined to be the next node in the lattice, except those directly above and below, as illustrated in Figure 1.

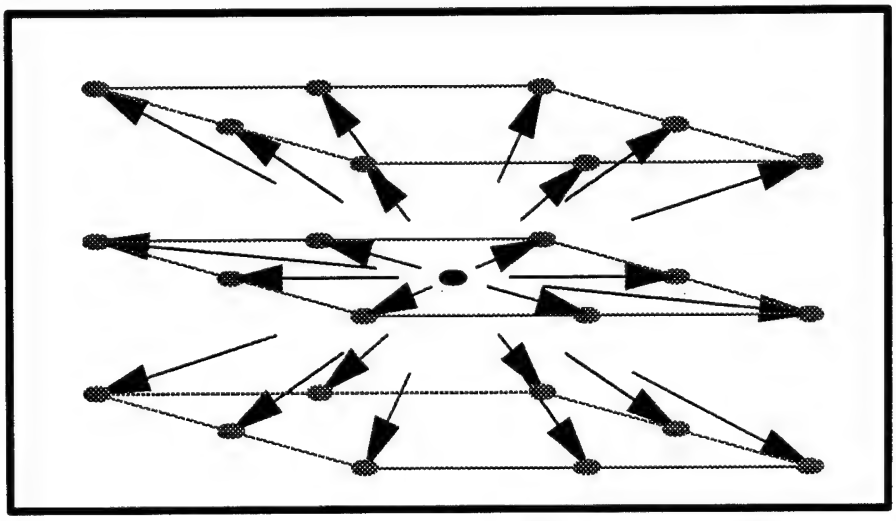

Figure 1. Nearest Neighbor to a Node.

This means that an aircraft is allowed to fly in any of these possible directions. Terrain avoidance is achieved by not constructing arcs that would pass through terrain. Depending on the terrain, this reduces the number of arcs substantially (15% in our example terrain).

### 2. Arc Costs

Attached to each arc are the fuel "costs" $t_{ij}$, and detection "costs" $c_{ij}$. In the solution of each relaxed sub-problem, a new Lagrange multiplier $\mu$ (Ahuja, et al., 1993, pp. 599) is used. The significance of $\mu$ is explained below. The "composite cost" for arc (i,j) is defined as $c_{ij} + \mu t_{ij}$ and is recomputed each time $\mu$ is changed.

In terms of the model, $\mu$ represents the weight assigned to the importance of the fuel constraint. If $\mu$ is zero, the fuel constraint is ignored in the network since the "composite cost" for arc (i,j) becomes $c_{ij}$. As $\mu$ increases, fuel "costs" become increasingly important as the value of $t_{ij}$ begins to dominate $c_{ij}$ for arc (i,j).

# III. MODEL IMPLEMENTATION

Model implementation (coding) forms a large part of this thesis. Selection of a suitable programming platform and language, as well as the appropriate algorithms and data structures are important. Algorithmic efficiency and the design of a suitable graphical interface are two main concerns.

## A. PROGRAMMING LANGUAGE AND PLATFORM

A Silicon Graphics (SGI) Onyx computer with 192 megabytes of memory was used in program development. Appendix A gives a detailed hardware configuration. The reason for using the SGI machine is that the machine is relatively fast for both numerical computations and graphical applications. The program is coded in C++ but does not use any object-oriented features of C++. The graphical interface is coded in OpenGL (McMinds, 1993) and Motif (Neider, 1993).

## B. ALGORITHMS AND DATA STRUCTURES

### 1. Hierarchical Adjacency List

The network is stored as a hierarchical adjacency list (HAL) in forward star form (Ahuja, et al., 1993, pp. 35-37). Each arc stores information on detection cost, fuel cost and other housekeeping information required by the label-correcting algorithm. In addition, a pointer is attached to

each node pointing to all out-going arcs from that node. This structure is chosen because the network remains unchanged once it is defined. Only arc costs are recomputed on each iteration. The HAL is efficiently implemented in a static array and has to accommodate a maximum of 24 times the maximum number of nodes.

## 2. Lagrangian Relaxation

Lagrangian relaxation can be used to incorporate the fuel consumption constraint into the objective function using a Lagrange multiplier $\mu$. The resultant relaxed shortest-path sub-problem can then be solved efficiently using a label-correcting shortest-path algorithm, discussed in the next section.

Since the original constrained problem is relaxed, the resultant solution may not be feasible. In the model, feasibility is obtained when:

$$\sum_{(i,j)\epsilon P} t_{ij} \leq T \qquad (1)$$

where P is the shortest-path for the relaxed sub-problem and T is the maximum fuel allowed for the problem.

It can be proven (Ahuja, et al., 1993, pp. 600-601) that

$$LB(\mu) = \sum_{(i,j)\epsilon P(\mu)} ( c_{ij} + \mu * t_{ij} ) - \mu T$$

is a lower bound on the length of the constrained shortest-path, where $P(\mu)$ is the shortest-path for the relaxed sub-problem.

The gap between the actual cost of the original constrained problem and the lower bound is defined as:

$$gap(\mu) = \sum_{(i,j)\epsilon P(\mu)} c_{ij} - LB(\mu) \qquad (2)$$

We can find a value of $\mu$ for the solution to the relaxed shortest-path sub-problem such that (1) holds and the gap specified by (2) is as small as possible. The basic idea is to employ a binary search to determine this "optimal" value of $\mu$. Starting with an initial value of $\mu$, the arc costs are computed as $c_{ij}' = c_{ij} + \mu t_{ij}$ and the relaxed shortest-path problem is solved. Depending on the value of the computed lower bound, and the feasibility of the solution, $\mu$ is either increased or decreased. The arc costs are then re-computed and process repeated. This procedure can be terminated either after a fixed number of iterations or when the feasible solution is reasonably close to optimal. Figure 2 shows the pseudocode for this algorithm.

Algorithm **Lagrangian Relaxation**;

Inputs:
  G = (N,A) Network in HAL format
  $c_{ij}$ Detection "costs"
  $t_{ij}$ Fuel "costs"
  T Fuel capacity
  $\mu_{max}$ Value of $\mu$ that ensures $P(\mu)$ will be feasible
  OptCr Stopping criteria

{

Define arc lengths, $c_{ij}' = c_{ij} + \mu * t_{ij}$, in network using $\mu = 0$;

Solve the relaxed shortest-path problem, to construct path P(0);

If $\sum_{(i,j)\epsilon P(0)} t_{ij} \leq T$

  goto Print;

else {

  Set $\mu_{lower} = 0$ and $\mu_{upper} = \mu_{max}$;
  Set iterations = 0;
  Set isOptimal = FALSE;

  While (iterations < max_iterations && not(isOptimal)) {

    iterations++;

    $\mu = (\mu_{upper} + \mu_{lower}) / 2$;

    Recompute arc lengths, $c_{ij}' = c_{ij} + \mu * t_{ij}$;

    Solve the shortest-path problem, with arc length $c_{ij}'$, giving path $P(\mu)$;

    Compute lower bound $LB(\mu) = \sum_{(i,j)\epsilon P(\mu)} (c_{ij} + \mu * t_{ij}) - \mu T$;

    If ($LB(\mu)$ > LB) LB = $LB(\mu)$;

Figure 2: Pseudocode for Lagrangian Relaxation Algorithm.

Figure 2 continued.

```
     If    Σ    t_ij  ≤ T
        (i,j)∈P(μ)

           μ_upper = μ;

           Save P(μ) as P;

     else

           μ_lower = μ;

     If ((   Σ    c_ij  - LB) < OptCr) isOptimal = TRUE;
           (i,j)∈P(μ)

    }      // end while

  }        // end if

}          // end algorithm

Print      "Near-optimal solution path is ";  P;
           "with value of μ of ";             μ;
           "Global lower bound is ";          LB;
           "Detection cost is ";              Σ    c_ij
                                           (i,j)∈P
           "Fuel cost is ";                   Σ    t_ij
                                           (i,j)∈P
```

Figure 2: Pseudocode for Lagrangian Relaxation Algorithm.

### 3. Shortest-path Sub-problem

The algorithm used to solve each relaxed shortest-path sub-problem is a version of a label-correcting algorithm which uses a dequeue (Double Ended Queue) (Ahuja, et al., 1993 pp. 141-143). A dequeue allows elements to be added both at the front and back of a queue. The shortest-path algorithm stores nodes to investigate on the dequeue which represent

potentially shorter paths. This algorithm is known to work very efficiently in practice although it does have exponential worst-case complexity.

A label-setting (Dijkstra's) algorithm (Ahuja, et al., 1993, pp. 108-112) was also implemented to validate results of the label-correcting algorithm in the developmental stages of the model.

# IV. TESTING AND EVALUATION

The Lagrangian relaxation algorithm was coded as described in the previous chapter. Testing and evaluation of this algorithm, and its shortest-path subroutine, is then achieved using a Graphical Evaluation Suite (GES). This chapter summarizes the methods and results of this testing.

## A. GRAPHICAL EVALUATION SUITE

A Graphical Evaluation Suite (GES) was built to facilitate the analysis and evaluation of the optimization model. GES is a graphical interface that displays computed routes, and allows the user to perform interactive analysis. This interface was designed for use by the model developer and is not meant for final use in operational systems. Appendix B briefly describes the human-machine interface (HMI) of GES. Appendix C provides a summary of important information pertaining to files and variables used by GES and AROMA.

## B. LABEL-CORRECTING ALGORITHM

Before integrating the label-correcting algorithm into the Lagrangian relaxation algorithm, the label-correcting algorithm was tested using several small example problems. These test problems were then checked against results obtained from the slower label-setting algorithm.

Next, several test cases in the actual network were set up using different radar databases. The label-correcting algorithm was tested on these cases for its speed and solution quality. Each iteration of the label-correcting algorithm on a network of more than 2 million arcs (200 x 200 x 3 nodes) is timed to take, on average, about four seconds. Validation is facilitated by GES which displays the solution path. Figures 3 to 6 show screen printouts of four of these cases.

Figure 3 illustrates an unconstrained solution that has many unnecessary turns. Figure 4 shows a solution where the fuel constraint is very important. Figures 5 and 6 show two near-optimal solutions that show that the label-correcting algorithm selects a "correct" solution. While problems with more than two radars were tested, the two-radar case is used as an example here to simplify the illustration.

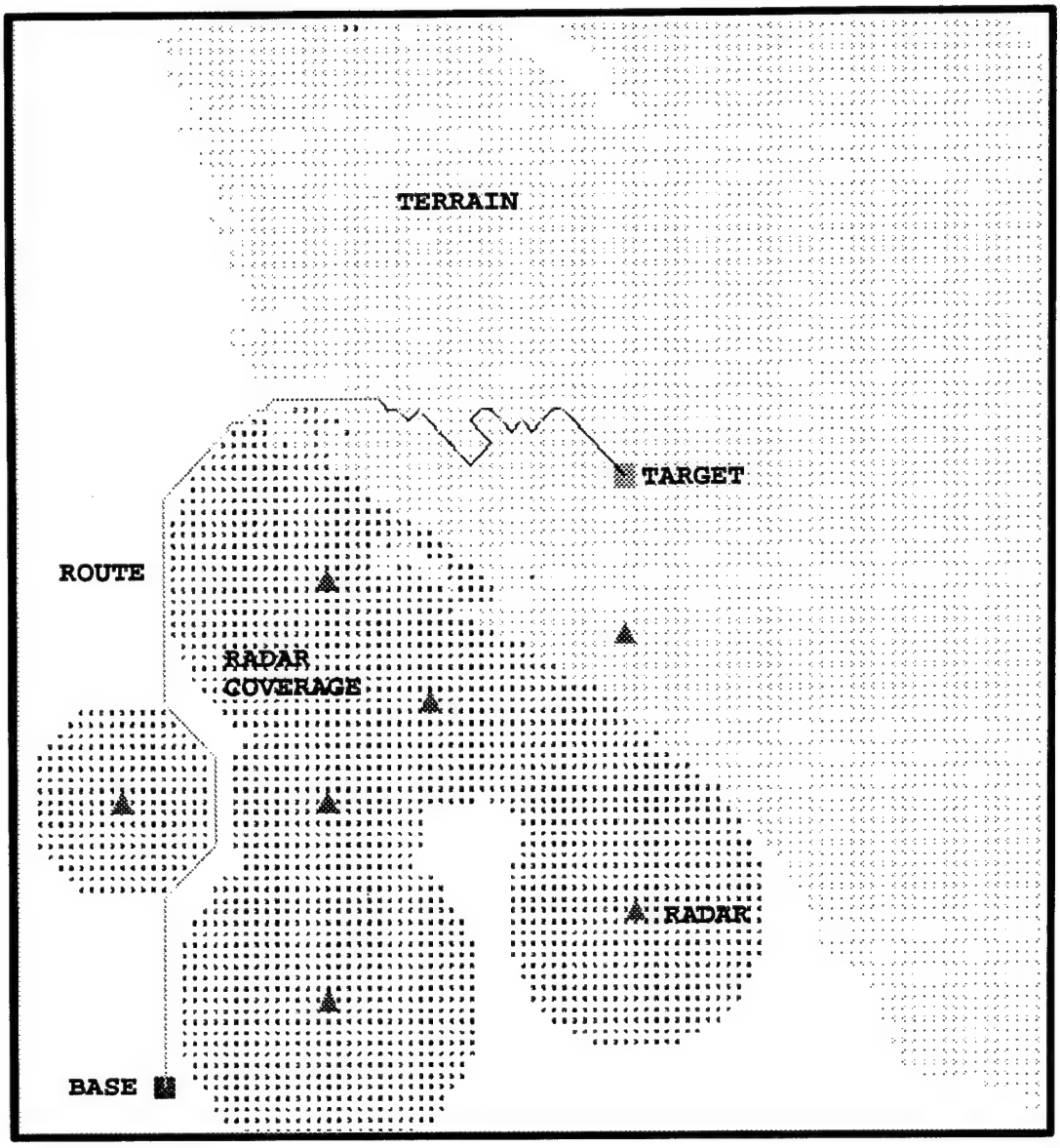


Figure 3. Typical Solution Route for $\mu = 0$.

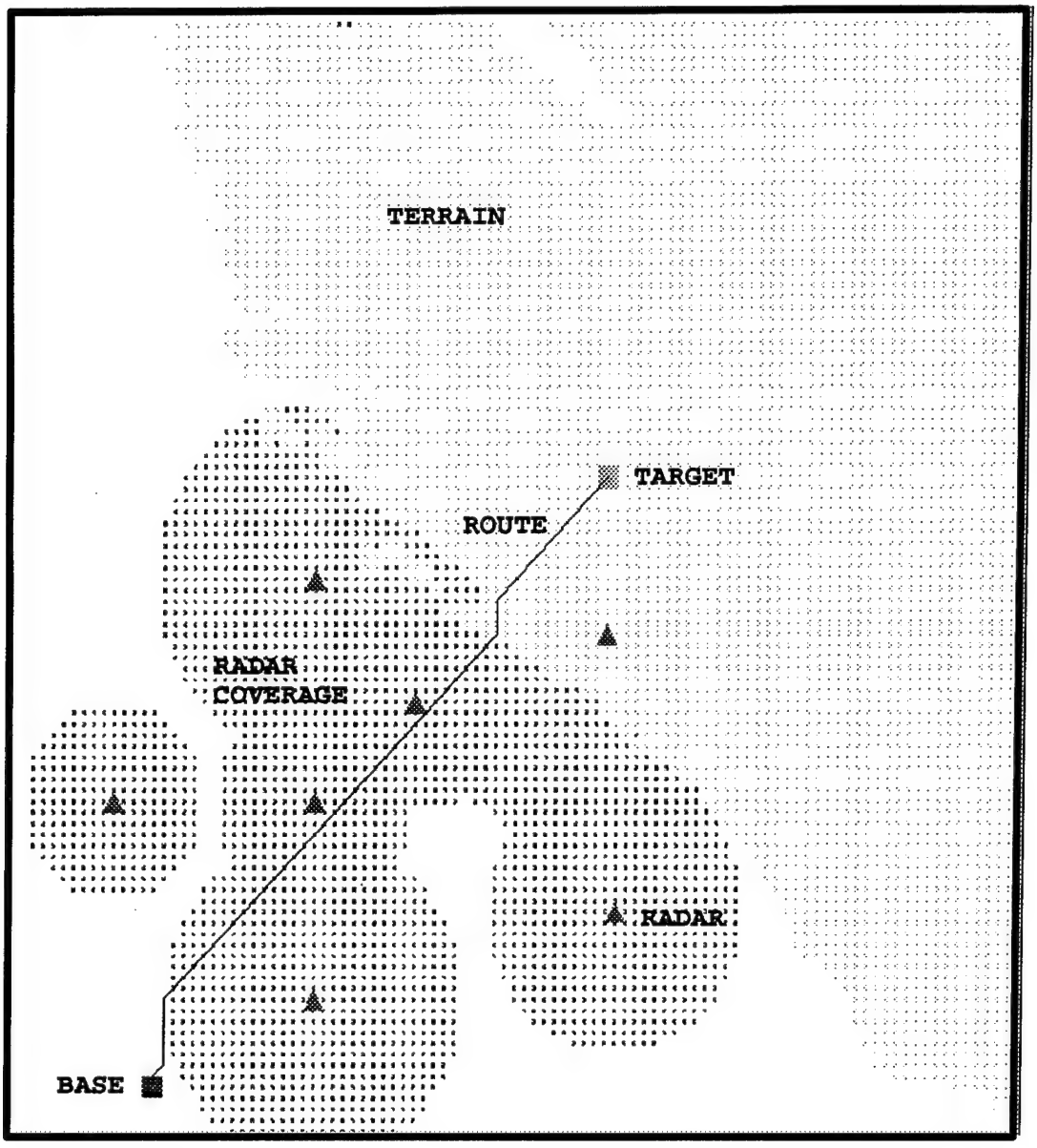


Figure 4. Typical Solution Route for Large $\mu$.

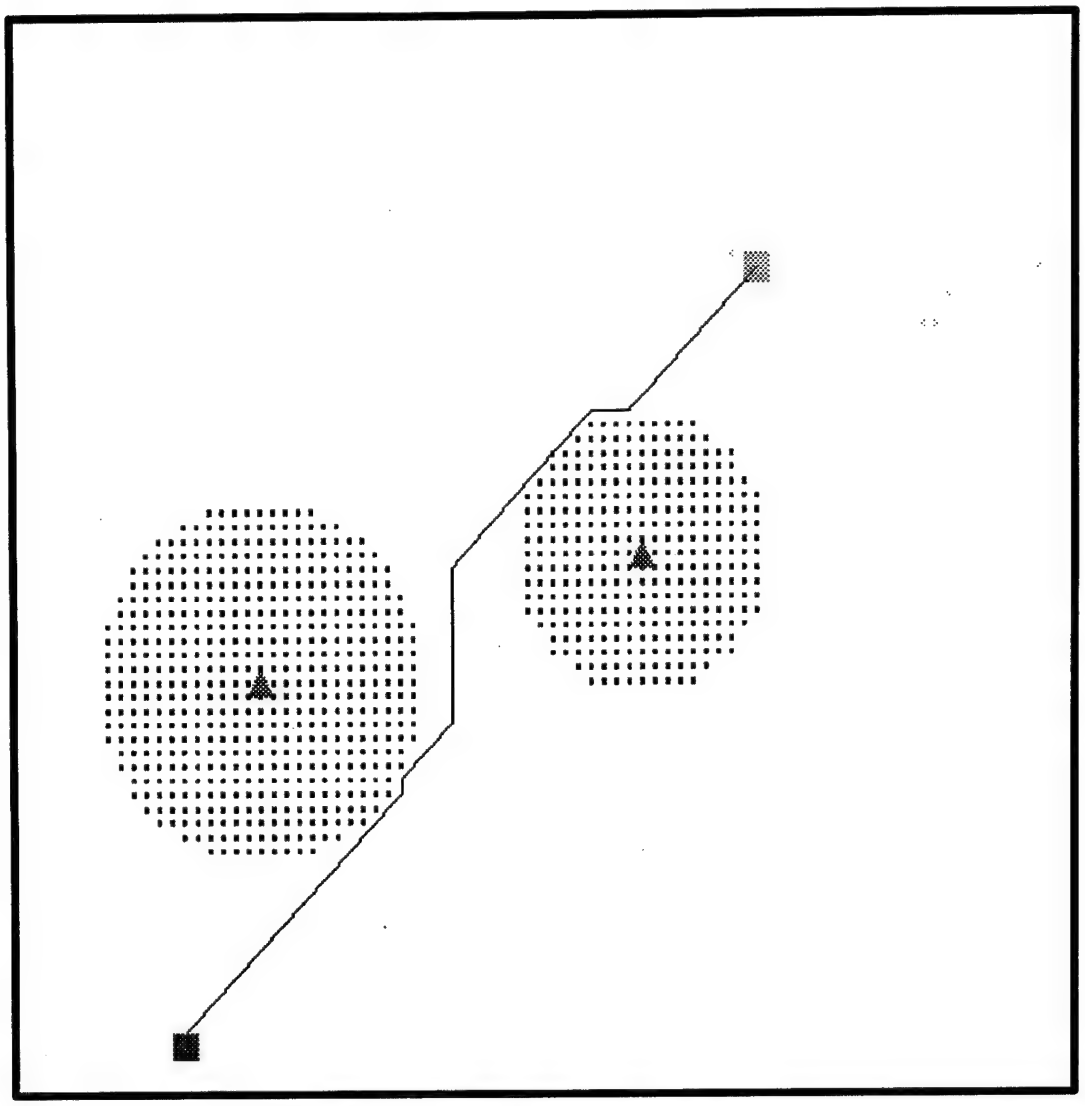

Figure 5. Typical Solution Route for $\mu$ Near-optimal for Two Non-overlapping Radars.

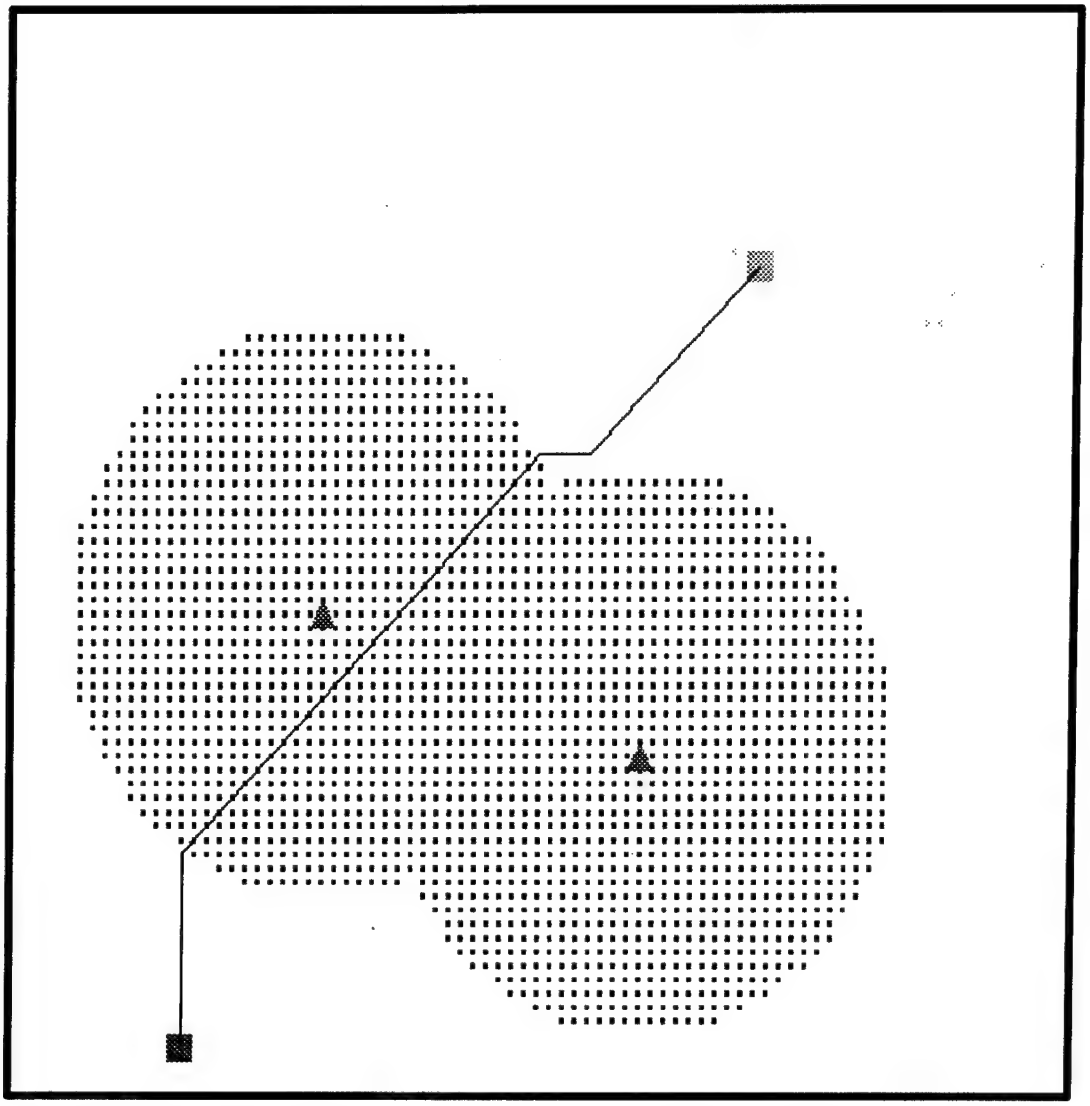

Figure 6. Typical Solution Route for $\mu$ Near-optimal for Two Overlapping Radars.

The discretization of the airspace introduces errors in the computation of distances and thus fuel consumption. Figure 7 shows a solution which clearly illustrates this point. Geometrically, the best solution is a straight line. However, due to the discretization, the solution uses two legs, one straight and the other diagonal to achieve the same effect. This results in extraneous fuel consumption and depending on the scenario, unnecessary exposure to radar detections.

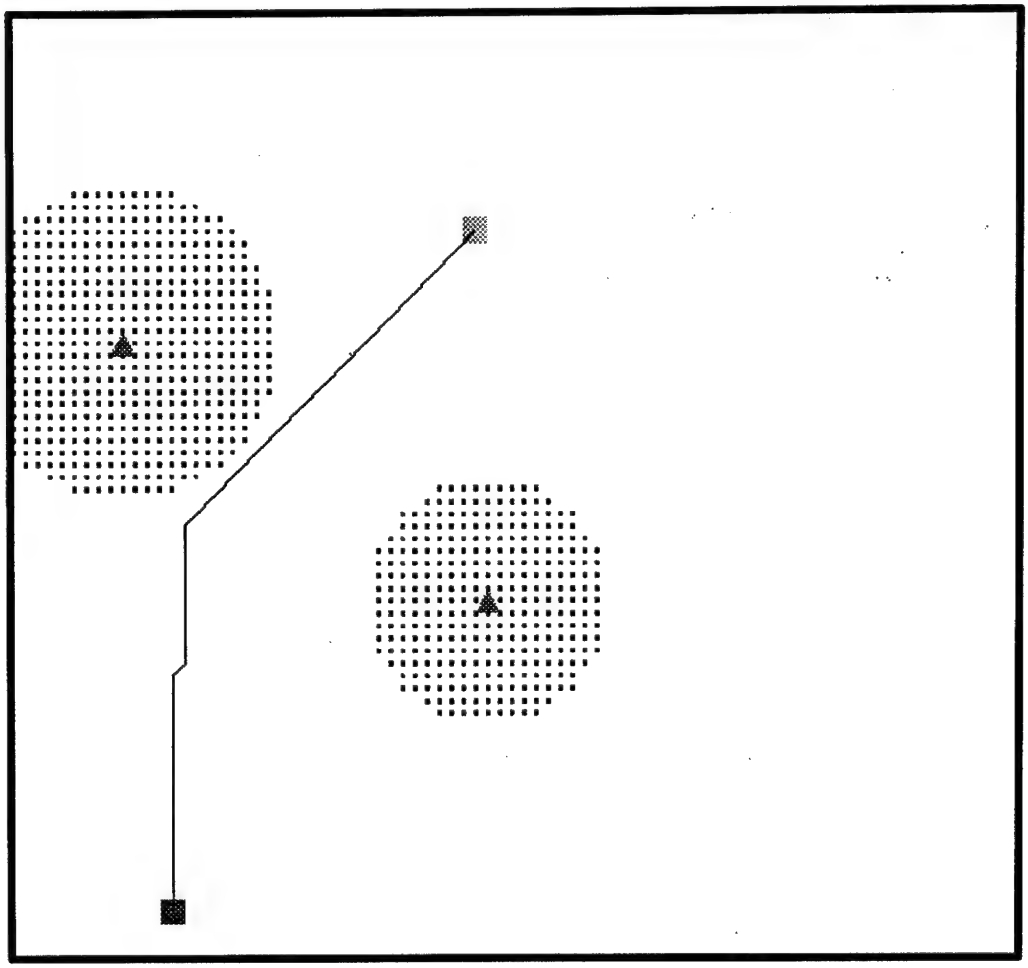

Figure 7. Illustration of Discretization Effect

## C. LAGRANGIAN RELAXATION

Before incorporating Lagrangian relaxation into the actual model, it was first tested on an example problem taken from Ahuja, et al., 1993, pp. 599-609. Table 1 contains the output generated by the algorithm and clearly shows that the algorithm works well and converges to the answer of 15 units after 21 iterations. It can also be observed that after about 11 iterations, the solution is already "close" to the known optimal with a percentage deviation from the lower bound of 114%. This deviation (optimality gap) is defined as

$$\left( \sum_{(i,j) \epsilon P(\mu)} c_{ij} - LB \right) / LB \qquad (3)$$

using the notation of Figure 2.

Maximum Fuel capacity: 14 units

| Iter | $\mu$ | Feasible | Lower Bound | Composite Cost | Detection Cost | Fuel Cost | % from Lower Bound |
|---|---|---|---|---|---|---|---|
| 1 | 5.00000 | Yes | -6.00000 | 64.00000 | 24.00 | 8.00 | -5.00 |
| 2 | 2.50000 | Yes | 5.00000 | 40.00000 | 15.00 | 10.00 | 2.00 |
| 3 | 1.25000 | No | 6.25000 | 23.75000 | 5.00 | 15.00 | -0.20 |
| 4 | 1.87500 | No | 6.87500 | 33.12500 | 5.00 | 15.00 | -0.27 |
| 5 | 2.18750 | Yes | 6.25000 | 36.87500 | 15.00 | 10.00 | 1.40 |
| 6 | 2.03125 | Yes | 6.87500 | 35.31250 | 15.00 | 10.00 | 1.18 |
| 7 | 1.95312 | No | 6.95312 | 34.29680 | 5.00 | 15.00 | -0.28 |
| 8 | 1.99219 | No | 6.99219 | 34.88285 | 5.00 | 15.00 | -0.28 |
| 9 | 2.01172 | Yes | 6.95312 | 35.11720 | 15.00 | 10.00 | 1.16 |
| 10 | 2.00195 | Yes | 6.99220 | 35.01950 | 15.00 | 10.00 | 1.15 |
| 11 | 1.99707 | No | 6.99707 | 34.95605 | 5.00 | 15.00 | -0.29 |
| 12 | 1.99951 | No | 6.99951 | 34.99265 | 5.00 | 15.00 | -0.29 |
| 13 | 2.00073 | Yes | 6.99708 | 35.00730 | 15.00 | 10.00 | 1.14 |
| 14 | 2.00012 | Yes | 6.99952 | 35.00120 | 15.00 | 10.00 | 1.14 |
| 15 | 1.99982 | No | 6.99982 | 34.99730 | 5.00 | 15.00 | -0.29 |
| 16 | 1.99997 | No | 6.99997 | 34.99955 | 5.00 | 15.00 | -0.29 |
| 17 | 2.00005 | Yes | 6.99980 | 35.00050 | 15.00 | 10.00 | 1.14 |
| 18 | 2.00001 | Yes | 6.99996 | 35.00010 | 15.00 | 10.00 | 1.14 |
| 19 | 1.99999 | No | 6.99999 | 34.99985 | 5.00 | 15.00 | -0.29 |
| 20 | 2.00000 | No | 7.00000 | 35.00000 | 5.00 | 15.00 | -0.29 |
| 21 | 2.00000 | Yes | 7.00000 | 35.00000 | 15.00 | 10.00 | 1.14 |

Table 1. Results of Test Run of Lagrangian Relaxation Algorithm.

The results of this test problem demonstrate that the algorithm works correctly, but that there can be a large optimality gap, as defined in (3), even when an optimal solution is found. Improving the "tightness" of the lower bound is a topic that will require further study.

After establishing correctness of the Lagrangian relaxation algorithm, testing was conducted on networks of sizes of from 0.5 million to 2.0 million arcs. Two key observations were made during testing. First, the algorithm gradually slows down as it approaches optimality. One possible explanation for this is that as the solution approaches optimality, there are more potential arcs to be considered in each sweep by the label-correcting algorithm, thus slowing the algorithm down. The second observation is that the final few solutions for a problem are often minor variations of one main theme. This can be clearly seen from Figure 8 which shows the best four solutions of a typical test run. These routes are all essentially the same with minor variations.

Table 2 shows the textual output of a typical test run used in producing solutions shown in Figure 8. The fuel capacity is 100 units. After 15 iterations, the value of $\mu$ is 0.41198 and the optimality gap is 305%. In this test run, the algorithm took about 75 seconds for 15 iterations.

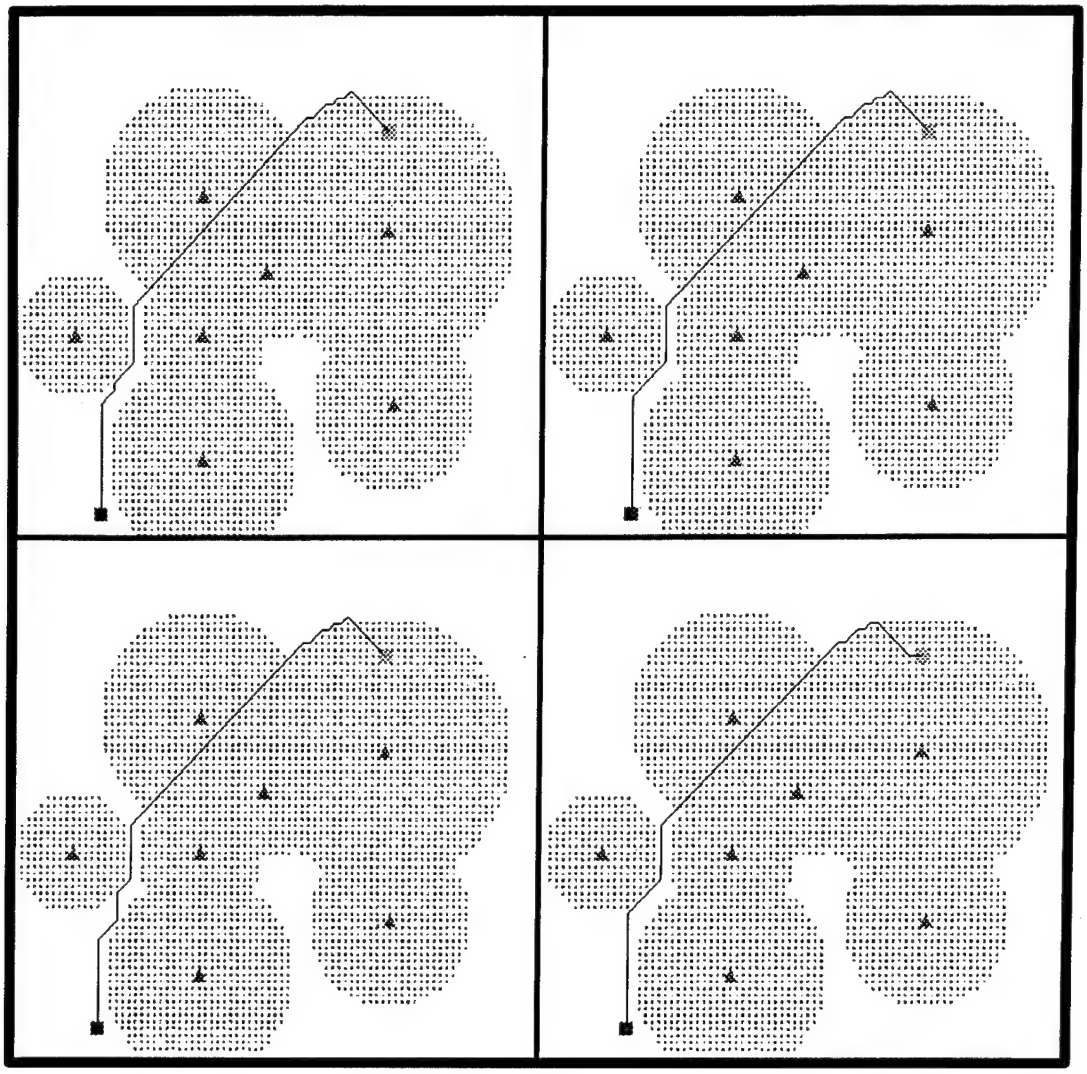

Figure 8. Best Four Solutions for a Typical Run.

Maximum Fuel Capacity : 100 units

| Iter | $\mu$ | Feasible | Lower Bound | Composite Cost | Detection Cost | Fuel Cost | % from Lower Bound |
|---|---|---|---|---|---|---|---|
| 1 | 50.00000 | Yes | -1256.63867 | 3743.36133 | 35.54 | 74.41 | -1.028 |
| 2 | 25.00000 | Yes | -616.99731 | 1883.00269 | 35.54 | 74.41 | -1.057 |
| 3 | 12.50000 | Yes | -297.17871 | 952.82129 | 35.54 | 74.41 | -1.119 |
| 4 | 6.25000 | Yes | -137.26984 | 487.73016 | 35.54 | 74.41 | -1.258 |
| 5 | 3.12500 | Yes | -57.31488 | 255.18512 | 35.54 | 74.41 | -1.620 |
| 6 | 1.56250 | Yes | -17.33739 | 138.91261 | 35.54 | 74.41 | -3.050 |
| 7 | 0.78125 | Yes | 1.45787 | 79.58287 | 26.88 | 76.75 | 17.439 |
| 8 | 0.39062 | No | 6.12828 | 45.19078 | 5.49 | 109.54 | -0.102 |
| 9 | 0.58393 | Yes | 4.18404 | 62.77779 | 26.57 | 87.34 | 5.350 |
| 10 | 0.48828 | Yes | 5.42002 | 54.24814 | 26.57 | 87.34 | 3.902 |
| 11 | 0.43945 | Yes | 6.00206 | 49.94737 | 25.65 | 88.17 | 3.274 |
| 12 | 0.41503 | Yes | 6.29083 | 47.79474 | 25.65 | 88.17 | 3.078 |
| 13 | 0.40283 | No | 6.24480 | 46.52801 | 5.49 | 109.54 | -0.119 |
| 14 | 0.40893 | No | 6.30303 | 47.19659 | 5.49 | 109.54 | -0.127 |
| 15 | 0.41198 | Yes | 6.32694 | 47.52567 | 25.65 | 88.17 | 3.055 |

Table 2. Textual Output of the Test Run used in Producing Solutions Shown in Figure 8.

# V. CONCLUSION

A prototypic automatic route optimization model was designed and implemented. This model is able to solve typical problems of (200 square nautical miles) in a reasonable amount of time (generally within two to three minutes). In addition, a Graphical Evaluation Suite (GES) was developed that allows the user to visualize the solutions easily and facilitates interactive analysis.

## A. OBSERVATIONS

The observations made during the testing and evaluation of AROMA are as follows:

### 1. Integrality Gap

The gap between a near-optimal feasible solution and its lower bound can be quite large. Most of the results produced by the model exhibit this large gap. However, from visual feedback from GES, these solutions can hardly be far from optimal. In the analysis of this gap, results from the test example shown in Table 1 was investigated. In this case where the known optimal solution was found, it was still 114% from the lower bound of seven units. This demonstrates that a good solution can be very far from its lower bound.

The reason for this gap is explained in Ahuja, et al., 1993, pp. 614-419). One possible reason why these gaps are large (greater than 300% in many cases in the problems tested) can be explained using Table 2. The numerical interpretation is that as $\mu$ approaches optimal, it appears that two potential routes with composite costs of about 47 are being evaluated. The first has a detection "cost" of 5.49 but is infeasible while the other has a detection "cost" of 25.65 but is feasible. The low detection "cost" of 5.49 for the infeasible solution is possible since there are many arcs in the network that have zero probability "cost". Depending on the value of $\mu$ and the scaling of $c_{ij}$ and $t_{ij}$, the algorithm is able to tradeoff detection "cost" for fuel "cost" for an approximately similar composite cost. The large difference between the feasible and infeasible "costs" accounts for the large integrality gap.

### 2. Solution Iterations

It was observed that the label-correcting algorithm slows down as the solution approaches optimal. If no iteration limit is imposed, there are situations where the model takes a large amount of time to reach an "optimal" solution. However, it was also observed that in most of the cases tested, a reasonably good solution can be achieved within 15 to 20 iterations. As illustrated by Figure 8, further iterations normally give solutions that are not

significantly different. It is recommended some investigation be made into the effect of the following factors on the number of iterations required: number of radars (consequently, the number of arcs with non-zero arc costs) and scaling of $c_{ij}$ and $t_{ij}$.

The heuristic selection of an appropriate value of $\mu_{max}$ (the maximum value that $\mu$ can take) will also help reduce unnecessary iterations. Currently, the binary search for optimal $\mu$ starts using a left and right limit. The left limit is zero and right limit is hard coded as $\mu_{max}$. Some heuristic method of intelligently choosing this value could save several iterations. $\mu_{max}$ must be large enough so that the optimal value of $\mu$ falls within the search interval and yet small enough so that unnecessary iterations are reduced.

### 3. Probablistic Assumption

To linearize the probability costs for each arc, logarithms of these costs (log costs) are used. The GES allows the user an option whether to use log costs. Tests comparing the solution using log costs with solutions without log costs, showed no significant difference in solution routes, although the solution costs differed. One possible reason for this effect is that given the constant probability of detection, and the data sets used, solutions can always be

found through areas where only one radar had coverage. Thus, each arc along the path would contribute a fixed unit cost regardless of whether the log was used or not. More investigation has to be done to analyze this effect.

#### 4. Discretization of Airspace

As explained earlier, the discretization of the airspace introduces errors in the computation of distances and thus fuel consumption. More work needs to be done to investigate the effect of discretization on the quality of solutions. If much better solutions can be found when there is no fuel wastage, then further research into a way to overcome this problem must be found. The effect of a smaller grid interval can also be investigated.

## B. AREAS FOR FUTURE RESEARCH

During the course of developing AROMA, there have been some assumptions made for the sake simplicity. Some of these simplifications are areas for future research to make AROMA a more robust model.

### 1. Radar Model

#### a. Assignment of Probability Densities

Using a constant probability of detection method to assign "probability mass" to nodes is just one method commonly used in planning. The main purpose for doing

that here was to concentrate attention on the solution algorithm itself. Conceptually, a more complicated method of assigning "probability mass" (the one described by Leary (1994) being an example) can be incorporated into the pre-processing step. However, the method must subscribe to the probability assumptions that have been made earlier.

### b. Time in Radar Coverage

The detection cost is constant for an arc regardless of its Euclidean length. This implies that time in coverage is not accurately considered. Some apportionment method can be derived to factor this into the arc costs computation methodology. One possible but simplistic method is to use as the arc cost, the product of the current arc cost $c_{ij}$ with the Euclidean length (with necessary scaling). This will make it more costly to traverse a longer arc than a shorter one if detection cost for the two are similar.

Arcs are assigned a probablity cost if the node from which it originates is within the radar detection region. Thus, some arcs that are partially in the region are assigned a cost while others are not. By using geometry, these arcs can be properly accounted for and some pro-rated costs be assigned accordingly.

### c. Radar Line-of-sight

Further development should be made to the radar model to include line-of-sight computations and atmospheric effects. Certainly, this would make detection of lower level flight more difficult and encourage more low-level flight paths. Currently, most of our solutions involve a large segment of high level flight because there is no incentive to select lower routes which consume more fuel but offer no significant reduction in detection costs.

## 2. Model Efficiency

### a. Improved Label-Correcting Algorithm

The label-correcting algorithm currently runs quite efficiently. However, further development can be made to improve its performance. The node at the front of the dequeue is always selected as the next candidate from which to extend the shortest path. One suggestion to improve the performance of the algorithm is to expend some effort selecting the "best" node (with minimum distance) from the dequeue rather than the first node. Selecting the absolutely best node from the dequeue would yield Dijkstra's algorithm which will typically be less efficient. However, expending a modest amount of effort to find a "better than first" node may yield improved efficiency without undue overhead.

### b. Defined Area of Operations

The system currently has provisions for the user to specify a more narrow area of operations. This can help eliminate large areas of search space, so that solution time can be reduced. Another possibility arising out of a smaller search space is a finer grid interval. However, the network setup routine does not now use this fact to eliminate nodes and arcs. Some vector algebra, similar to that done in computer graphics algorithms (Foley, 1990) to determine if a point is in a closed region, can be done to determine if nodes, and consequently arcs are within the defined area of operations.

## 3. Operational Implementation

### a. Route Smoothing

The discretization of the airspace causes some jaggedness in the solution route. One possible improvement would be to do post-processing smoothing of the solution route without increasing the fuel costs incurred. A plausible method that can be considered is as follows. First, a geometrical volume smaller than the original search space can be used to envelop the solution path. Next, a new network can be constructed by connecting nodes to their nearest neighbors as before, but also to all nodes within a specified distance beyond the nearest neighbors. Then, a shortest-path algorithm can be used to select a "smoother" route from this network.

Feasibilty in the original problem is assured since geometrically, the original path forms an upper bound.

Another alternative to achieve smoothing is to employ techniques used in computer graphics, e.g., splines or Bezier curves (Foley, 1990). However, these routes will generally not be straight.

### b. Return Leg

Operational route optimization has to consider both the attack and return phases of the mission. The model only considers the attack phase. From the modelling perspective, the main difference between the two phases is the change of data, for example, fuel capacity and fuel consumption rates (since load is either released or jettisoned). From the operational perspective, there are other considerations. For example, the early flight segments of the return phase should typically avoid the last flight segments of the attack phase within a defined perimeter. One of the reasons that this is done is to ensure de-confliction between incoming and outgoing planes in the target area.

### c. Designated Waypoints/Taboo Areas

The model should incorporate the ability to accept locations in the network through which aircraft must fly through, for example, re-fuelling points. For a single re-fuelling point, the problem can be considered as two

distinct sub-problems.

There are also areas that must be avoided, for example, populated areas. This can be achieved by assigning large positive costs to the taboo nodes or simply eliminating these nodes all together, as with nodes occurring within terrain.

### 4. Graphical Evaluation Suite (GES)

Although the current human-machine interface (GES) can already provide reasonable visual feedback and certainly facilitates interactive analysis, it is far from fully operational. Some of the data currently hard-coded in the program should be set interactively. Routes generated on each iteration of the algorithm could be displayed "on-the-fly" so that the user can visually appreciate how solutions are arrived at and make some intuitively assessment of the quality of the solutions.

## APPENDIX A : DETAILED HARDWARE CONFIGURATION

Silicon Graphics Onyx Computer

4 x 150 MHZ IP19 Processors

CPU: MIPS R4400 Processor Chip Revision: 5.0

FPU: MIPS R4010 Floating Point Chip Revision: 0.0

Data cache size: 16 Kbytes

Instruction cache size: 16 Kbytes

Secondary unified instruction/data cache size: 1 Mbyte

Main memory size: 192 Mbytes, 1-way interleaved

## APPENDIX B : GRAPHICAL EVALUATION SUITE (GES)

### 1. Display Legend

The graphical display on the left of the window shows all pertinent spatial information e.g., terrain, location of radars, their detection envelopes and routes. The legend for the display is as follows:

| Item | Description |
|---|---|
| Homebase | Green square |
| Target | Red square |
| Route | Colored line joining homebase to target. Different flight levels are distinguished by different colors, i.e. a change of color implies a change of flight level. |
| Radars | Triangles |
| $P_d$ envelopes | Yellow areas (darker => higher $P_d$) |
| Terrain | Blue areas (for a particular display height) |
| Map | Gray areas when map is turned on (lighter => higher altitude) |

### 2. Textual Outputs

Textual output is printed in the command window and to an output file "results.out". The details of this file are explained in Appendix C.

### 3. Control Panel

On the right of the window is the control panel. Description of the functionality of the controls are as follows:

| | |
|---|---|
| **Toggle Buttons** | |
| Boundary | Not implemented. |
| Branching | Not implemented. |
| Log Cost | Toggles whether log detection costs are used in the construction of the network. |
| Map | Displays color coded map of underlying terrain. |
| **Solutions** | If a optimization run is made, selecting P1 to P5 will display the five best routes generated. |
| **Action Buttons** | |
| Solve LC | Solves a single iteration of the network using label-correcting algorithm. A value of $\mu$ is read from file "radar.dat" (see Appendix B for description of file) |
| Read In | Reads updated inputs from "radar.dat". This has to be done whenever radar locations, maximum fuel, toll need to be changed. |
| Clear | Clears path display. |
| Optimize | Run the optimization algorithm. Output is saved into "results.out". |
| Exit | Ends program. |
| **Display Level** | Since the graphical display is two dimensional, this slider allows the user to see different height levels. |

## APPENDIX C. PROGRAM FILES AND VARIABLES

**radar.dat** File used to read in program data. This file contains the following information (and format):

| Field | Format | Comments |
|---|---|---|
| maximum fuel | float | maximum fuel for mission. |
| toll | float | Lagrangian multiplier used in single iteration label- correcting algorithm run. |
| number of radars | integer | number of radars in the database. The next rows of data contain the following information per row. |
| radar number | integer | radar ID which is not used in the program. |
| x position | integer | x position of radar in our area of 200 by 200 nautical mile. |
| y position | integer | y position of radar. |
| minimum range | float | minimum range of radar for our defined probability of detection. |
| maximum range | float | maximum range of radar for our defined probability of detection. |

**elevation.dat** Terrain file generated by the pre-processing program *printht.c* which collates several DTED files into this 200 nautical mile by 200 nautical mile format. This file is only changed if the area of operation changes. The first two data are the number of points in the x and y dimension. The next two are the minimum and maximum height in the datafile. The rest of the data points are integer elevations for the area in row order.

**results.out** This file contains results generated from an optimization run given the data in radar.dat. The top row shows the maximum fuel allowed. Each subsequent row is formatted as follows:

| Field | Comments |
|---|---|
| Iter | Iteration count. |
| Toll | Toll value (Lagrangian multiplier) used in this iteration. |
| Feasible | Indicates whether solution is feasible. |
| Lower bound | Lower bound for the current iteration. |
| Composite cost | Actual cost of the relaxed sub-problem. |
| Cost | Actual detection cost for current solution ignoring contribution of constraint. |
| Fuel | Fuel costs for current solution. |
| Percentage | Percentage away from lower bound of current solution. |

**Variables** There are several variables which are important for the execution of the model. These are summarized below.

| Name | Location | Comments |
|---|---|---|
| HT_BASE<br>HT_SEP | radar.h | base height and separation for lattice. |
| Target.x<br>Target.y<br>Home.x<br>Home.z | planner.C | target and home base location. |
| PD_CURVE | radar.h | probability value of equal probability region. |
| MAX_X<br>MAX_Y<br>MAX_Z | radar.h | maximum dimensions of lattice which is currently set to 200 x 200 x 3. |

## LIST OF REFERENCES

[1] Leary, John J. III, "Search for a Stealthy Flight Path Through a Radar Defense Network", Master's Thesis, Naval Postgraduate School, December 1994.

[2] Boerman, A. Douglas, "Finding an Optimal Path Through a Mapped Minefield", Master's Thesis, Naval Postgraduate School, March 1994.

[3] Ong, Seow Meng, "A Mission Planning Expert System with Three-Dimensional Path Optimization for NPS Model 2 Autonomous Underwater Vehicle", Master's Thesis, Naval Postgraduate School, June 1990.

[4] Wrenn, Lawrence R. III, "Three-Dimensional Route Planning for a Cruise Missile for Minimal Detection by Observer", Master's Thesis, Naval Postgraduate School, June 1989.

[5] Ahuja, Ravindra K., Magnanti, Thomas L., Orlin, Jams B., Network Flows, Englewoods Cliffs, New Jersey: Prentice-Hall Inc., 1993.

[6] Aho, Alfred V., Hopcroft, John E., Ullman, Jeffrey D., Data Structures and Algorithms, Reading, Pennsylvania: Addison-Wesley Publishing Company, 1983.

[7] McMinds, Donald L., Mastering OSF/Motif Widgets, Reading, Massachusetts: Addision-Wesley, 1993.

[8] Neider, Jackie, OpenGL, Reading, Massachusetts: Addison-Wesley, 1993.

[9] Foley, James D., Computer Graphics: Principles and Practices, Reading, Massachusetts: Addison-Wesley, 1990.

## INITIAL DISTRIBUTION LIST

| | | No. copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304-6145 | 2 |
| 2. | Library, Code 52<br>Naval Postgraduate School<br>Monterey, CA 93943-5101 | 2 |
| 3. | Professor R. Kevin Wood<br>Department of Operations Research<br>Naval Postgraduate School, Code OR/Wd<br>Monterey, CA 93943-5008 | 8 |
| 4. | Professor Gordon H. Bradley<br>Department of Operations Research<br>Naval Postgraduate School, Code OR/BZ<br>Monterey, CA 93943-5008 | 2 |
| 5. | Cpt. Steve H. K. Lee<br>c/o Professor R. Kevin Wood<br>Department of Operations Research<br>Naval Postgraduate School, Code OR/Wd<br>Monterey, CA 93943-5008 | 2 |